Tries

Specialized and misspelled trees

Sample problem : how to store a phone directory

06 00 00 00 00 : John

06 00 22 45 86 : Dany

06 01 21 45 67 : Cersei

06 01 22 43 68 : Sansa

06 02 43 88 88 : Arya

07 10 11 12 13 : Night King

Sample problem : how to store a phone directory

06 00 00 00 00 : John

06 00 22 45 86 : Dany

06 01 21 45 67 : Cersei

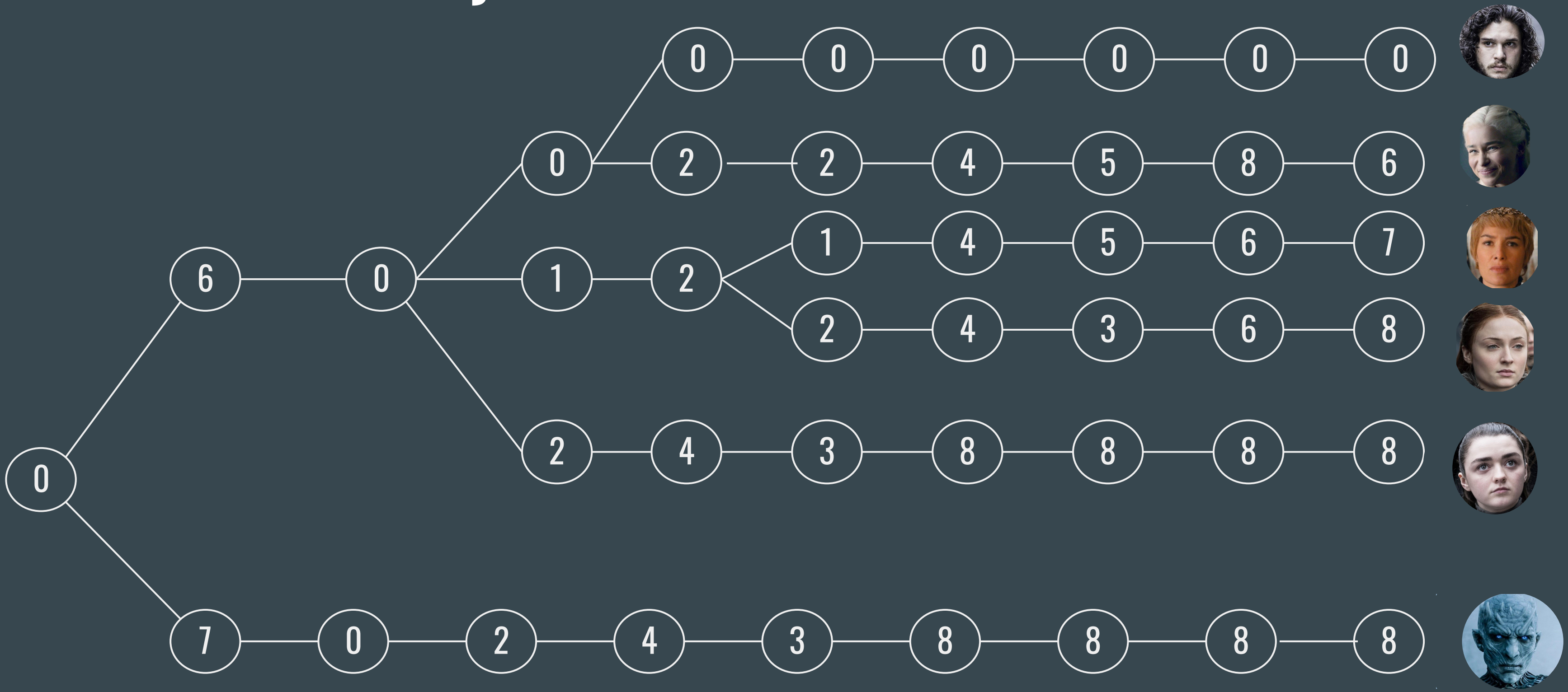
06 01 22 43 68 : Sansa

06 02 43 88 88 : Arya

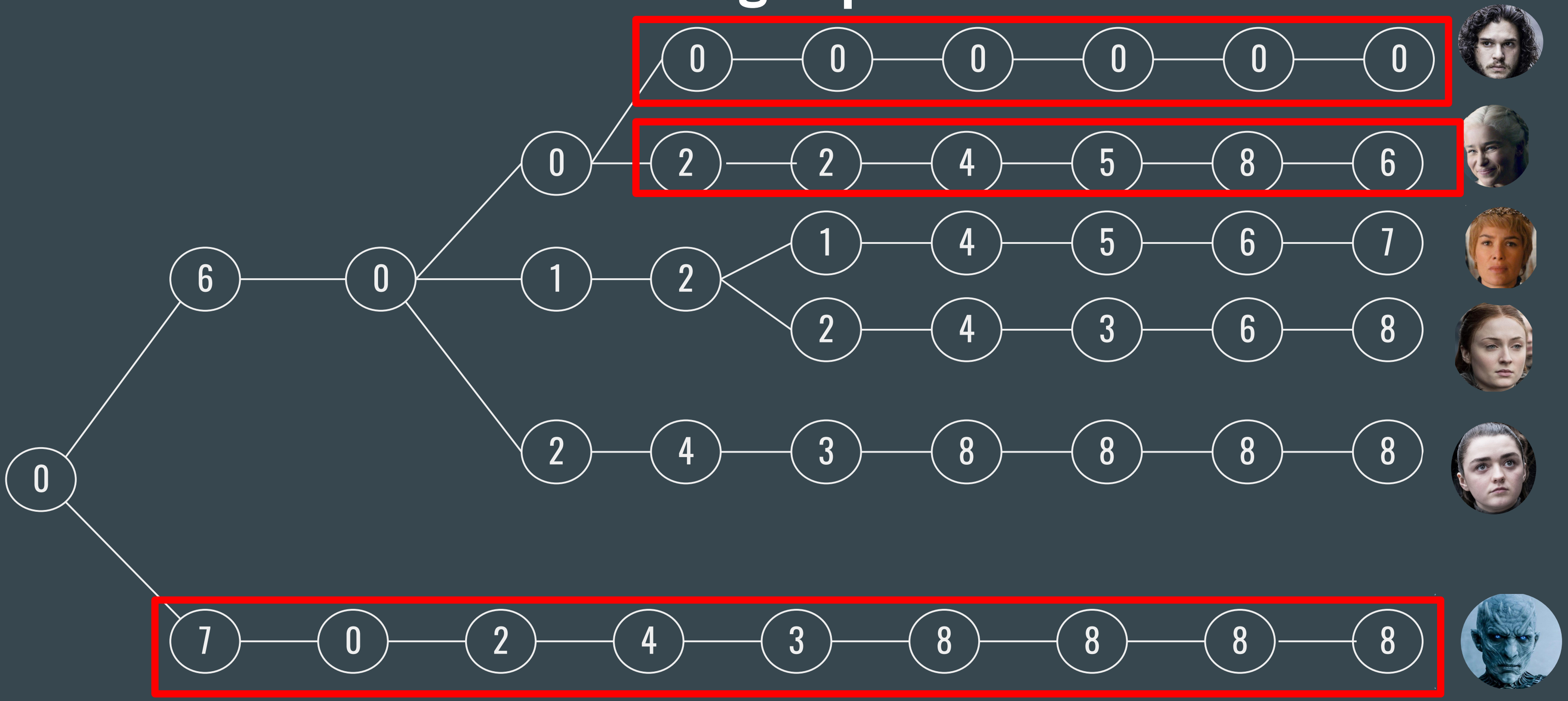
07 10 11 12 13 : Night King

Do we really need that information 5 times?

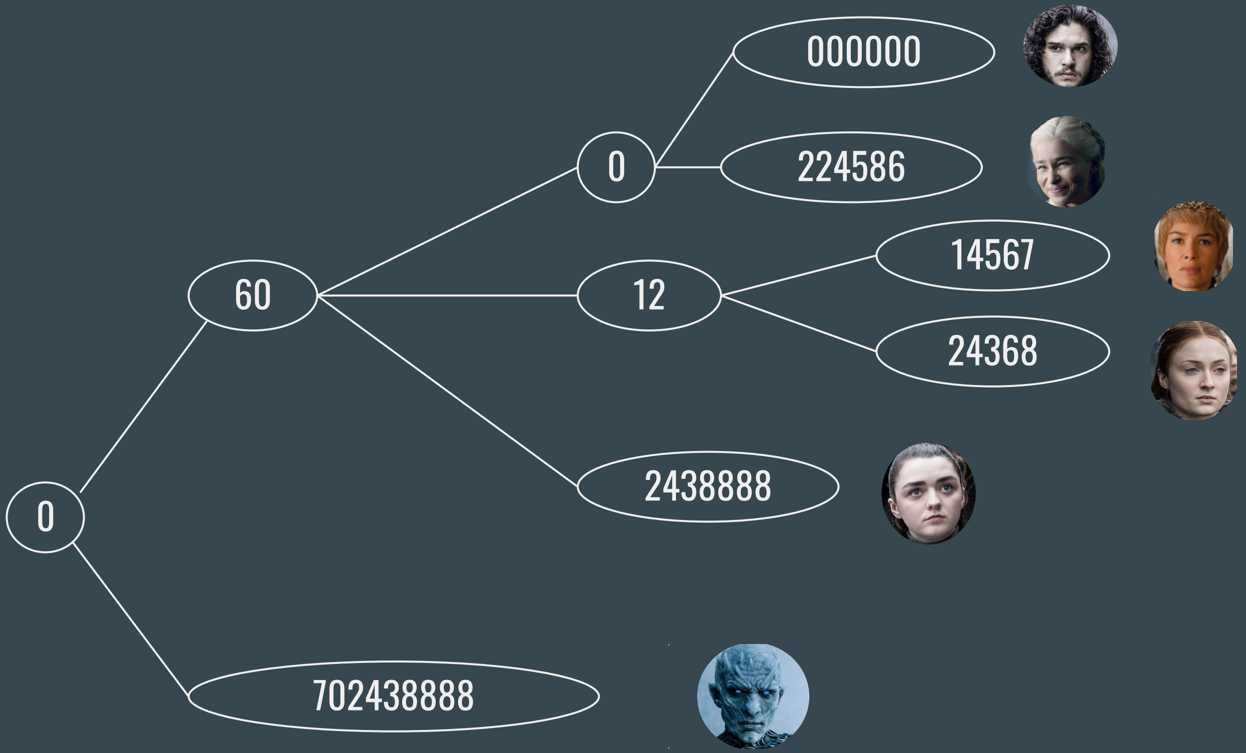
Much better that way !



That’s a lot of nodes for a straight path



A compressed version



Prefix & suffix tries

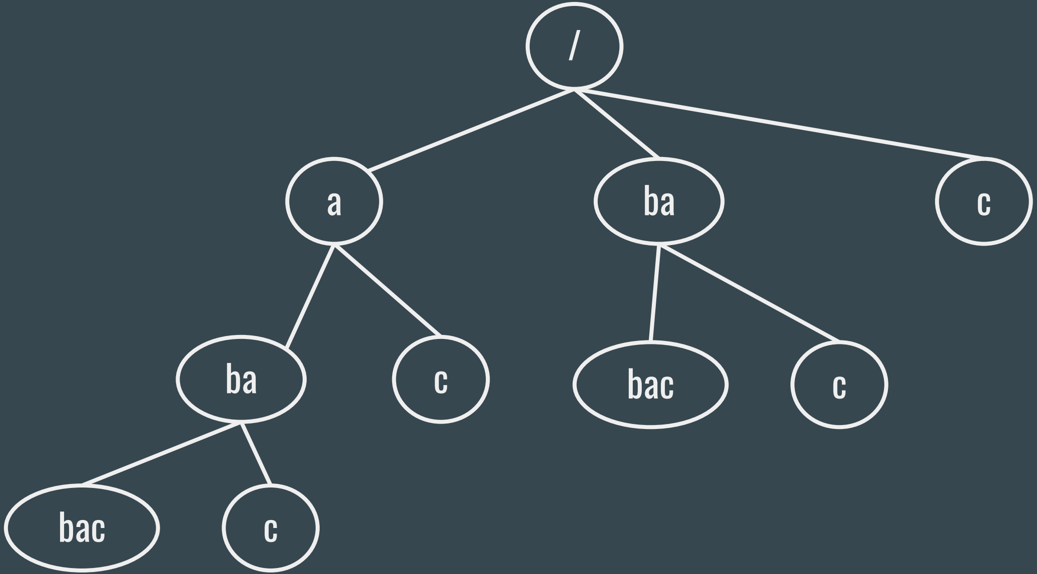
- Same principle of construction, but this time the elements of the collection are prefixes/suffixes of a string

What is a suffix?

- A substring that contains the last character of a string

“ababac” → { “c”, “ac”, “ bac”, “abac”, “babac”, “ababac”}

Suffix tree for “ababac”



“ababac” → { “c”, “ac”, “ bac”, “abac”, “babac”, “ababac”}

How to build a suffix tree?

With n the size of the string the tree represents :

- Naive implementation runs in $O(n^2)$
  - Generate every suffix ($O(n)$), then insert them into the tree ($O(n^2)$)
- An algorithm exists to build them in $O(n)$!
  - Ukkonen Algorithm
  - “The proof is left to the reader”!

What are tries effective for?

- Data compression, auto completion
- String manipulation
    - Longest palindromic substring
    - Pattern searching
    - Longest substring common to a set of strings
    - Shortest unique substring
    - Many more !

Credits

Author : Arthur Tondereau

Sources :

Wikipedia : https://en.wikipedia.org/wiki/Suffix_tree

https://en.wikipedia.org/wiki/Trie

THE Algorithm design MANUAL, 12.3 . S.S. Skiena